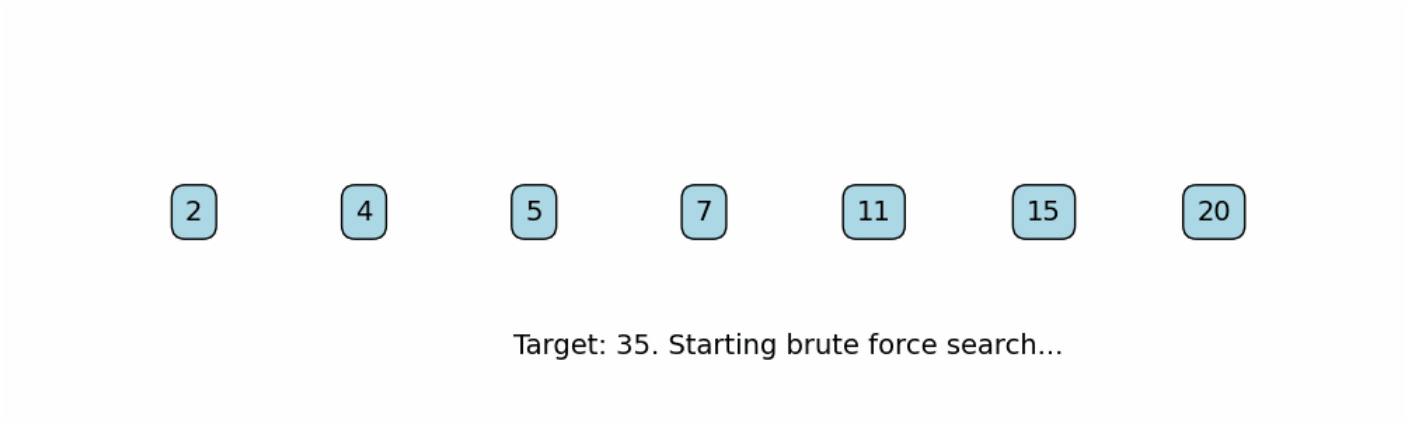
Brute Force

The brute force approach tries every possible combination to check for a solution, without leveraging any special properties or optimizations of the data (such as sorted order). Typically involves:

- Nested loops
  - Outer Loop traverses the array for the first element of the pair
  - Inner Loop traverses the rest of the array to find second element
- Recursion

Drawbacks:

- **Time Complexity:** Typically O(n^2) or slower, impractical for large datasets.
- **Redundancy:** Many computations are repeated unnecessarily




```
def brute_force_two_sum(nums, target):
    # Iterate through each element in the list
    for i in range(len(nums)):
        # For each element, check every other element that comes after it
        for j in range(i + 1, len(nums)):
            # Check if the current pair sums to the target
            if nums[i] + nums[j] == target:
                return (i, j)  # Return the indices as a tuple
    # If no pair is found that adds up to the target, return None
    return None

# Example usage:
nums = [2, 7, 11, 15]
```

```
target = 9

result = brute_force_two_sum(nums, target)
if result:
    print("Pair found at indices:", result)
else:
    print("No pair found that adds up to the target.")
```

Revision #4
Created 2 January 2025 22:30:38 by victor
Updated 9 February 2025 23:29:43 by victor